

Bound Your Models!

How to Make OWL an ASP Modeling Language

Sarah Alice Gaggl, Sebastian Rudolph, Lukas Schweizer

Technische Universität Dresden

Abstract. To exploit the Web Ontology Language OWL as an answer set programming (ASP) language, we introduce the notion of bounded model semantics, as an intuitive and computationally advantageous alternative to its classical semantics. We show that a translation into ASP allows for solving a wide range of bounded-model reasoning tasks, including satisfiability and axiom entailment but also novel ones such as model extraction and enumeration. Ultimately, our work facilitates harnessing advanced semantic web modeling environments for the logic programming community through an “off-label use” of OWL.

Keywords: Answer Set Programming, Bounded-Model Semantics, Semantic Web

1 Introduction

Answer set programming (ASP) is a powerful declarative language for knowledge representation and reasoning [4]. In ASP the knowledge is encoded in a set of logical rules and interpreted under the *stable model semantics* [8,9]. Recent developments led to powerful systems e.g. dlv [17], and gringo/clasp [6], to name some of them, which are capable to solve a large variety of problems [7]. In particular, ASP has shown to be well suited for big combinatorial search problems, as the dedicated solvers are specially designed to enumerate all solutions [2].

However, it has often been noted that, while being a powerful and versatile formalism, popularity and widespread adoption of logic programming in general and answer set programming in particular is hindered by the non-availability of user-friendly and scalable editing environments.

On the other side, formalisms coming with a more elaborate modeling tool support – most notably the Web Ontology Language OWL [31] – are often preferred, even if the application scenario actually is of a constraint-satisfaction type which does not go well with OWL’s standard semantics allowing for models of arbitrary size. Ontology editors like Protégé [16] provide user-friendly interfaces and combined with the natural language alike Manchester syntax [12] possesses perspicuous access to a presumably complex and involved formalism.

We propose *bounded model reasoning* as an intuitive and simple approach to overcome this situation. Thereby, we endow OWL with a non-standard model-theoretic semantics and modifying the modelhood condition by restricting the

domain to a finite set of bounded size, induced by the named individuals occurring in the given OWL ontology. We note that this additional condition can be axiomatized in the latest version of OWL. While reasoning in OWL under the classical semantics is N2EXPTIME-complete [15], we show that reasoning under the *bounded model semantics* is merely NP-complete. Still, employing the axiomatization, existing OWL reasoners struggle on bounded model reasoning, due to the heavy combinatorics involved.

Therefore, we propose a different approach and define a translation of *SR_{OIQ}* knowledge bases (the logical counterparts to OWL ontologies) into answer set programs [4], such that the set of bounded models coincides with the set of answer sets of the obtained program, allowing us to use existing answer set solvers (see [2] for an overview) for bounded model reasoning. Next to the inferencing tasks typically used in semantic web technologies, this approach also allows for solving other, non-standard reasoning problems like model enumeration.

The benefits are manifolded, whereas in this work we particularly emphasize OWL as modeling language for typical constraint-satisfaction-type problems. The translation based approach can be seen as higher-level layer on top of the ASP language. Although we focus on the description logic *SR_{OIQ}* and its native DL syntax, other syntax specifications like the OWL 2 Manchester Syntax [12] very well strive towards user-friendliness by means of natural language features.

We have implemented the proposed approach, for which first preliminary evaluations on typical constraint-satisfaction-type problems not only demonstrate feasibility, but also suggest significant improvement compared to the axiomatized approach using highly optimized OWL reasoners.

The article is organized as follows. In Section 2 we introduce the necessary background on description logics and ASP. Then, in Section 3 we define the *bounded model semantics* and analyze their complexity. The particular encoding of a *SR_{OIQ}* knowledge base into ASP is given in Section 4. A preliminary evaluation of the implemented system is summarized in Section 5. Finally, we conclude in Section 6 and discuss possible future directions.

2 Preliminaries

In this section we provide the necessary background of description logics and answer set programming.

2.1 Description Logics

OWL 2 DL, the version of the Web Ontology Language we focus on, is defined based on description logics (DLs, [3,25]). We briefly recap the description logic *SR_{OIQ}* (for details see [13]). Let N_I , N_C , and N_R be finite, disjoint sets called *individual names*, *concept names* and *role names* respectively. These atomic entities can be used to form complex ones as displayed in Table 1. A *SR_{OIQ}* knowledge base is a tuple $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ where \mathcal{A} is a *SR_{OIQ}* ABox, \mathcal{T} is a *SR_{OIQ}*

Table 1. Syntax and semantics of role and concept constructors in \mathcal{SROIQ} , where a_1, \dots, a_n denote individual names, s a role name, r a role expression and C and D concept expressions.

Name	Syntax	Semantics
inverse role	s^-	$\{(x, y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (y, x) \in s^{\mathcal{I}}\}$
universal role	u	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top	\top	$\Delta^{\mathcal{I}}$
bottom	\perp	\emptyset
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
nominals	$\{a_1, \dots, a_n\}$	$\{a_1^{\mathcal{I}}, \dots, a_n^{\mathcal{I}}\}$
univ. restriction	$\forall r.C$	$\{x \mid \forall y.(x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
exist. restriction	$\exists r.C$	$\{x \mid \exists y.(x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
<i>Self</i> concept	$\exists r.Self$	$\{x \mid (x, x) \in r^{\mathcal{I}}\}$
qualified number	$\leq n r.C$	$\{x \mid \#\{y \in C^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}}\} \leq n\}$
restriction	$\geq n r.C$	$\{x \mid \#\{y \in C^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}}\} \geq n\}$

Table 2. Syntax and semantics of \mathcal{SROIQ} axioms.

Axiom α	$\mathcal{I} \models \alpha$, if	
$r_1 \circ \dots \circ r_n \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \dots \circ r_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$	RBox \mathcal{R}
$\text{Dis}(s, r)$	$s^{\mathcal{I}} \cap r^{\mathcal{I}} = \emptyset$	
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	TBox \mathcal{T}
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	ABox \mathcal{A}
$r(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$	
$a \doteq b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$	
$a \not\equiv b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$	

TBox and \mathcal{R} is a \mathcal{SROIQ} RBox. Table 2 presents the respective axiom types available in the three parts, and we will refer to each TBox axiom as *general concept inclusion* (GCI). The original definition of \mathcal{SROIQ} contained more RBox axioms (expressing transitivity, (a)symmetry, (ir)reflexivity of roles), but these can be shown to be syntactic sugar. Moreover, the definition of \mathcal{SROIQ} contains so-called *global restrictions* which prevents certain axioms from occurring together. These complicated restrictions, while crucial for the decidability of classical reasoning in \mathcal{SROIQ} are not necessary for the bounded-model reasoning considered here, hence we omit them for the sake of brevity.

The semantics of \mathcal{SROIQ} is defined via interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ composed of a non-empty set $\Delta^{\mathcal{I}}$ called the *domain of \mathcal{I}* and a function $\cdot^{\mathcal{I}}$ mapping individual names to elements of $\Delta^{\mathcal{I}}$, concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This mapping is extended to complex role and concept expressions (cf. Table 1) and finally used to define satisfaction of axioms (see Table 2). We say that \mathcal{I} satisfies a knowledge base $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ (or \mathcal{I} is a model of \mathcal{K} , written: $\mathcal{I} \models \mathcal{K}$) if it satisfies all axioms of \mathcal{A} , \mathcal{T} , and \mathcal{R} . We say

that a knowledge base \mathcal{K} *entails* an axiom α (written $\mathcal{K} \models \alpha$) if all models of \mathcal{K} are models of α .

2.2 Answer-Set Programming

We give a brief overview of the syntax and semantics of disjunctive logic programs under the answer-sets semantics [9]. We fix a countable set \mathcal{U} of (*domain*) *elements*, also called *constants*; and suppose a total order $<$ over the domain elements. An *atom* is an expression $p(t_1, \dots, t_n)$, where p is a *predicate* of arity $n \geq 0$ and each t_i is either a variable or an element from \mathcal{U} . An atom is *ground* if it is free of variables. $B_{\mathcal{U}}$ denotes the set of all ground atoms over \mathcal{U} . A (*disjunctive*) *rule* ρ is of the form

$$a_1 \vee \dots \vee a_n \leftarrow b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m,$$

with $n \geq 0$, $m \geq k \geq 0$, $n + m > 0$, where $a_1, \dots, a_n, b_1, \dots, b_m$ are atoms, or a *count expression* of the form $\#count\{l : l_1, \dots, l_i\} \bowtie u$, where l is an atom and $l_j = p_j$ or $l_j = \text{not } p_j$, for p_j an atom, $1 \leq j \leq i$, u a non-negative integer, and $\bowtie \in \{\leq, <, =, >, \geq\}$. Moreover, “*not*” denotes *default negation*. The *head* of ρ is the set $H(\rho) = \{a_1, \dots, a_n\}$ and the *body* of ρ is $B(\rho) = \{b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m\}$. Furthermore, $B^+(\rho) = \{b_1, \dots, b_k\}$ and $B^-(\rho) = \{b_{k+1}, \dots, b_m\}$. A rule ρ is *normal* if $n \leq 1$ and a *constraint* if $n = 0$. A rule ρ is *safe* if each variable in ρ occurs in $B^+(\rho)$. A rule ρ is *ground* if no variable occurs in ρ . A *fact* is a ground rule with empty body and no disjunction. An (*input*) *database* is a set of facts. A program is a finite set of rules. For a program Π and an input database D , we often write $\Pi(D)$ instead of $D \cup \Pi$. If each rule in a program is normal (resp. ground), we call the program normal (resp. ground).

For any program Π , let U_{Π} be the set of all constants appearing in Π . $Gr(\Pi)$ is the set of rules $\rho\sigma$ obtained by applying, to each rule $\rho \in \Pi$, all possible substitutions σ from the variables in ρ to elements of U_{Π} . For count-expressions, $\{l : l_1, \dots, l_n\}$ denotes the set of all ground instantiations of l , governed through $\{l_1, \dots, l_n\}$. An *interpretation* $I \subseteq B_{\mathcal{U}}$ *satisfies* a ground rule ρ iff $H(\rho) \cap I \neq \emptyset$ whenever $B^+(\rho) \subseteq I$, $B^-(\rho) \cap I = \emptyset$, and for each contained count-expression, $N \bowtie u$ holds, where N is the cardinality of the set of ground instantiations of l , $N = |\{l \mid l_1, \dots, l_n\}|$, for $\bowtie \in \{\leq, <, =, >, \geq\}$ and u a non-negative integer. I satisfies a ground program Π , if each $\rho \in \Pi$ is satisfied by I . A non-ground rule ρ (resp., a program Π) is satisfied by an interpretation I iff I satisfies all groundings of ρ (resp., $Gr(\Pi)$). $I \subseteq B_{\mathcal{U}}$ is an *answer set* of Π iff it is a subset-minimal set satisfying the *Gelfond-Lifschitz reduct* $\Pi^I = \{H(\rho) \leftarrow B^+(\rho) \mid I \cap B^-(\rho) = \emptyset, \rho \in Gr(\Pi)\}$. For a program Π , we denote the set of its answer sets by $\mathcal{AS}(\Pi)$.

3 Bounded Models

When reasoning in description logics, models can be of arbitrary cardinality. In many applications, however, the domain of interest is known to be finite. In

fact, restricting DL reasoning to models of finite domain size (called *finite model reasoning*, a natural assumption in database theory), has become the focus of intense studies lately [18,5,24].

As opposed to assuming the domain to be merely finite (but of arbitrary, unknown size), we consider the case where the domain has an *a priori known cardinality*, more precisely, when the domain coincides with the set of named individuals mentioned in the knowledge base. We refer to such models as *bounded models* and argue that in many applications this modification of the standard DL semantics represents a more intuitive definition of what is considered and expected as *model* of some knowledge base.¹

Definition 1 (Bounded-Model Semantics). *Let \mathcal{K} be a \mathcal{SROIQ} knowledge base. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is said to be individual-bounded w.r.t. \mathcal{K} , if all of the following holds:*

1. $\Delta^{\mathcal{I}} = \{a \mid a \in N_I(\mathcal{K})\}$,
2. for each individual $a \in N_I(\mathcal{K})$, $a^{\mathcal{I}} = a$.

Accordingly, we call an interpretation \mathcal{I} (individual-)bounded model of \mathcal{K} , if \mathcal{I} is an individual-bounded interpretation w.r.t. \mathcal{K} and $\mathcal{I} \models \mathcal{K}$ holds. A knowledge base \mathcal{K} is called bounded-model-satisfiable if it has a bounded model. We say \mathcal{K} bounded-model-entails an axiom α (written $\mathcal{K} \models_{\text{bm}} \alpha$) if every bounded model of \mathcal{K} is also a model of α .

Note that, under the bounded-model semantics, there is a one-to-one correspondence between (bounded) interpretations and sets of ground facts, if one assumes the set of domain elements fixed and known. That is, for every bounded-model interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, we find exactly one Abox $\mathcal{A}_{\mathcal{I}}$ with atomic concept assertions and role assertions defined by $\mathcal{A}_{\mathcal{I}} := \{r(a, b) \mid (a, b) \in r^{\mathcal{I}}\} \cup \{A(a) \mid a \in \Delta^{\mathcal{I}}\}$ and likewise, every such Abox \mathcal{A} gives rise to a corresponding interpretation $\mathcal{I}_{\mathcal{A}}$. This allows us to use ABoxes as representations of models.

We briefly demonstrate the effects of bounded model semantics as opposed to finite model semantics (with entailment \models_{fin}) and the classical semantics. Let $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ with $\mathcal{A} = \{A(a), A(b), s(a, b)\}$, $\mathcal{T} = \{\top \sqsubseteq \exists r.B, \top \sqsubseteq \leq 1 r^-. \top\}$, and $\mathcal{R} = \{\text{Dis}(s, r)\}$. First we note that \mathcal{K} has a bounded (hence finite) model \mathcal{I} representable as $\mathcal{A}_{\mathcal{I}} = \{A(a), A(b), B(a), B(b), s(a, b), r(a, a), r(b, b)\}$, thus \mathcal{K} is satisfiable under all three semantics. Then $\alpha = \top \sqsubseteq \exists r. \exists r. B$ holds in all models of \mathcal{K} , therefore $\mathcal{K} \models \alpha$, $\mathcal{K} \models_{\text{fin}} \alpha$, and $\mathcal{K} \models_{\text{bm}} \alpha$. Opposed to this, $\beta = \top \sqsubseteq B$ merely holds in all finite models, whence $\mathcal{K} \models_{\text{fin}} \beta$ and $\mathcal{K} \models_{\text{bm}} \beta$, but $\mathcal{K} \not\models \beta$. Finally, $\gamma = \top \sqsubseteq \exists r. \text{Self}$ only holds in all bounded models, thus $\mathcal{K} \models_{\text{bm}} \gamma$, but $\mathcal{K} \not\models_{\text{fin}} \gamma$ and $\mathcal{K} \not\models \gamma$.

¹ In fact, when working with practitioners performing modeling tasks in OWL, we often found this to be their primary intuition, and OWL to be “abused” as a constraint language for an underlying fixed domain.

3.1 Extraction & Enumeration of Bounded Models

When performing satisfiability checking in DLs (the primary reasoning task considered there), a model constructed by a reasoner merely serves as witness to claim satisfiability, rather than an accessible artifact. However, as mentioned before, our approach aims at scenarios where a knowledge base is a formal problem description for which each model represents one solution. Then, retrieval of one, several, or all models is a natural task, as opposed to merely checking existence. With *model extraction* we denote the task of materializing an identified model in order to be able to work with it, i.e. to inspect it in full detail and reuse it in downstream processes. The natural continuation of model extraction is to make all models explicit, performing *model enumeration*. Conveniently, for both tasks we can use the introduced model representation via ABoxes.

Most existing DL reasoning algorithms attempt to successively construct a model representation of a given knowledge base. However, most of the existing tableaux reasoners do not reveal the constructed model, besides the fact that in the non-bounded case models might end up being infinite such that an explicit representation is impossible. Regarding enumeration, we state that this task is not supported – not even implicitly – by any state-of-the-art DL reasoner, also due to the reason that in the non-bounded case, the number of models is typically infinite and even uncountable. We want to stick to the notions of model extraction and enumeration as their meaning should be quite intuitive. Although, in the general first-order case the term *model expansion* is used, e.g. in the work of Mitchell and Ternovska [19]. There, an initial (partial) interpretation representing a problem instance is expanded to ultimately become a model for the encoded problem.

3.2 Complexity of Bounded Model Reasoning

The combined complexity of reasoning in *SR_{OLQ}* over arbitrary interpretations is known to be N2EXPTIME-complete [15]. Still, it is considered to be usable in practice since worst-case knowledge bases would be of very artificial nature. Restricting to bounded models leads to a drastic drop in complexity.

Theorem 1. *The combined complexity of checking bounded-model satisfiability of *SR_{OLQ}* knowledge bases is NP-complete.*

Proof. (Sketch) To show membership, we note that after guessing an interpretation \mathcal{I} , (bounded) modelhood can be checked in polynomial time. For this we let \mathcal{C} contain all the concept expressions occurring in \mathcal{K} (including subexpressions). Furthermore, let \mathcal{R} contain all role expressions and role chains (including subchains) occurring in \mathcal{K} . Obviously, \mathcal{C} and \mathcal{R} are of polynomial size. Then, in a bottom-up fashion, we can compute the extension $C^{\mathcal{I}}$ of every element C of \mathcal{C} and the extension $r^{\mathcal{I}}$ of every element r of \mathcal{R} along the defined semantics. Obviously, each such computation step requires only polynomial time. Finally, based on the computed extensions, every axiom of \mathcal{K} can be checked – again in polynomial time.

To show hardness, we note that any 3SAT problem can be reduced to bounded-model satisfiability as follows: Let $\mathcal{L} = \{L_1, \dots, L_n\}$ be a set of 3-clauses. Then satisfiability of $\bigwedge_{\{\ell_1, \ell_2, \ell_3\} \in \mathcal{L}} (\ell_1 \vee \ell_2 \vee \ell_3)$ coincides with the bounded-model satisfiability of the knowledge base containing the two axioms $\top(a)$ and $\top \sqsubseteq \prod_{\{\ell_1, \ell_2, \ell_3\} \in \mathcal{L}} (C_{\ell_1} \sqcup C_{\ell_2} \sqcup C_{\ell_3})$, where $C_{\ell_i} = A_p$ if $\ell_i = p$ and $C_{\ell_i} = \neg A_p$ if $\ell_i = \neg p$ for any propositional symbol p .

Note that this finding contrasts with the observation that bounded-model reasoning in first-order logic is PSPACE-complete. We omit the full proof here, just noting that membership and hardness can be easily shown based on the fact that checking modelhood in FOL is known to be PSPACE-complete [30] and, for the membership part, keeping in mind that NPSpace=PSPACE thanks to Savitch's Theorem [28]. This emphasizes the fact that, while the bounded-model restriction turns reasoning in FOL decidable, restricting to *SRQIQ* still gives a further advantage in terms of complexity (assuming $P \neq NP$).

3.3 Axiomatization of Bounded Models inside *SRQIQ*

When introducing a new semantics for some logic, it is worthwhile to ask if existing reasoners can be used. Indeed, it is easy to see that, assuming $\{a_1, \dots, a_n\} = N_I(\mathcal{K})$, adding the *SRQIQ* GCI $\top \sqsubseteq \{a_1, \dots, a_n\}$ as well as the set of inequality axioms containing $a_i \not\approx a_j$ with $i < j$ to \mathcal{K} will rule out exactly all the non-bounded models of \mathcal{K} . Denoting these additional axioms with \mathcal{BM} , we then find that \mathcal{K} is bounded-model satisfiable iff $\mathcal{K} \cup \mathcal{BM}$ is satisfiable under the classical DL semantics and, likewise, $\mathcal{K} \models_{\text{bm}} \alpha$ iff $\mathcal{K} \cup \mathcal{BM} \models \alpha$ for any axiom α . Consequently, any off-the-shelf *SRQIQ* reasoner can be used for bounded-model reasoning, at least when it comes to the classical reasoning tasks.

However, the fact that the currently available DL reasoners are not optimized towards reasoning with axioms of the prescribed type (featuring disjunctions over potentially large sets of individuals) and that available reasoners do not support model extraction and model enumeration led us to develop an alternative computational approach based on ASP.

4 Encoding *SRQIQ* Knowledge Bases into ASP

We propose an encoding of an arbitrary *SRQIQ* knowledge base \mathcal{K} , into an *answer set program* $\Pi(\mathcal{K})$, such that the set of answer sets $\mathcal{AS}(\Pi(\mathcal{K}))$, coincides with the set of bounded models of the given knowledge base. This allows us to use existing ASP machinery to perform both standard reasoning as well as *model extraction* and *model enumeration* quite elegantly. Intuitively, the set of all bounded models defines a search space, which can be traversed searching for models, guided by appropriate constraints. We thus propose an ASP encoding consisting of a generating part $\Pi_{\text{gen}}(\mathcal{K})$, defining all potential candidate interpretations, and a constraining part $\Pi_{\text{chk}}(\mathcal{K})$, ruling out interpretations violating the knowledge base.

Table 3. Ω -Normalization of knowledge base axioms.

$\Omega(\mathcal{K}) = \bigcup_{\alpha \in \mathcal{R} \cup \mathcal{A}} \Omega(\alpha) \cup \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \Omega(\top \sqsubseteq \text{nnf}(\neg C_1 \sqcup C_2))$	
$\Omega(\top \sqsubseteq \mathbf{C} \sqcup C') = \Omega(\top \sqsubseteq \mathbf{C} \sqcup \alpha_{C'}) \cup \bigcup_{\substack{1 \leq i \leq n \\ \text{for } C' \text{ of the form } C' = C_1 \sqcap \dots \sqcap C_n \text{ and } n \geq 2}} \Omega(\top \sqsubseteq \dot{\neg} \alpha_{C'} \sqcup C_i)$	
$\Omega(\top \sqsubseteq \mathbf{C} \sqcup \forall r.D) = \Omega(\top \sqsubseteq \mathbf{C} \sqcup \forall r.\alpha_D) \cup \Omega(\top \sqsubseteq \dot{\neg} \alpha_D \sqcup D)$	
$\Omega(\top \sqsubseteq \mathbf{C} \sqcup \geq n r.D) = \Omega(\top \sqsubseteq \mathbf{C} \sqcup \geq n r.\alpha_D) \cup \Omega(\top \sqsubseteq \dot{\neg} \alpha_D \sqcup D)$	
$\Omega(\top \sqsubseteq \mathbf{C} \sqcup \leq n r.D) = \Omega(\top \sqsubseteq \mathbf{C} \sqcup \leq n r.\dot{\neg} \alpha_{\dot{\neg} D}) \cup \Omega(\top \sqsubseteq \dot{\neg} \alpha_{\dot{\neg} D} \sqcup \dot{\neg} D)$	
$\Omega(\top \sqsubseteq \mathbf{C} \sqcup \neg\{s\}) = \begin{cases} \perp & \text{if } \mathbf{C} = \emptyset, \\ \Omega(\mathbf{C}(s)) & \text{otherwise.} \end{cases}$	
$\Omega(D(s)) = \{\alpha_D(s)\} \cup \Omega(\top \sqsubseteq \dot{\neg} \alpha_D \sqcup \text{nnf}(D))$	
$\Omega(r^-(s, t)) = \{r(t, s)\}$	
$\Omega(r_1 \circ \dots \circ r_n \sqsubseteq r) = \{r_1 \circ r_2 \sqsubseteq r_{(r_1 \circ r_2)}\} \cup \Omega(r_{(r_1 \circ r_2)} \circ r_3 \circ \dots \circ r_n \sqsubseteq r)$	
$\Omega(\beta) = \{\beta\} \text{ for any other axiom } \beta$	
$\alpha_C = \begin{cases} Q_C & \text{if } \text{pos}(C) = \text{true} \\ \neg Q_C & \text{if } \text{pos}(C) = \text{false} \end{cases}, \text{ where } Q_C \text{ is a fresh concept name unique for } C$	
$\begin{array}{ll} \text{pos}(\top) = \text{false} & \text{pos}(\perp) = \text{false} \\ \text{pos}(A) = \text{true} & \text{pos}(\neg A) = \text{false} \\ \text{pos}(\{s\}) = \text{true} & \text{pos}(\neg\{s\}) = \text{false} \\ \text{pos}(\exists r.\text{Self}) = \text{true} & \text{pos}(\neg\exists r.\text{Self}) = \text{false} \\ \text{pos}(C_1 \sqcap C_2) = \text{pos}(C_1) \vee \text{pos}(C_2) & \text{pos}(C_1 \sqcup C_2) = \text{pos}(C_1) \vee \text{pos}(C_2) \\ \text{pos}(\forall r.C_1) = \text{pos}(C_1) & \text{pos}(\leq n r.C_1) = \begin{cases} \text{pos}(\dot{\neg} C_1) & \text{if } n = 0 \\ \text{true} & \text{otherwise} \end{cases} \\ \text{pos}(\geq n r.C_1) = \text{true} & \end{array}$	
<p>Note: A is an atomic concept, $C_{(i)}$ are arbitrary concept expressions, \mathbf{C} is a possibly empty disjunction of concept expressions, D is not a literal concept. The function $\dot{\neg}$ is defined as $\dot{\neg}(\neg A) = A$ and $\dot{\neg}(A) = \neg A$ for some atomic concept A.</p>	

Our translation into ASP requires a knowledge base in normal form which can be obtained by an easy syntactic transformation.

Definition 2 (Normalized Form [22]). A GCI is normalized, if it is of the form $\top \sqsubseteq \bigsqcup_{i=1}^n C_i$, where C_i is of the form B , $\{a\}$, $\forall r.B$, $\exists r.\text{Self}$, $\neg\exists r.\text{Self}$, $\geq n r.B$, or $\leq n r.B$, for B a literal concept, r a role, and n a positive integer. A TBox \mathcal{T} is normalized, if each GCI in \mathcal{T} is normalized. An ABox \mathcal{A} is normalized if each concept assertion in \mathcal{A} contains only a literal concept, each role assertion in \mathcal{A} contains only an atomic role, and \mathcal{A} contains at least one assertion. An RBox \mathcal{R} is normalized, if each role inclusion axiom is of the form $r \sqsubseteq r'$ or $r_1 \circ r_2 \sqsubseteq r'$. A *SRIOIQ* knowledge base $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ is normalized if \mathcal{A} , \mathcal{T} , and \mathcal{R} are normalized.

Given $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$, the normalized form $\Omega(\mathcal{K})$ is obtained by applying a transformation Ω , given in Table 3, which is mainly standard in DLs [22]. The normalized knowledge base $\Omega(\mathcal{K})$ is a model-conservative extension of \mathcal{K} , i.e. every (bounded) model of $\Omega(\mathcal{K})$ is a (bounded) model of \mathcal{K} and every (bounded) model of \mathcal{K} can be turned into a (bounded) model of $\Omega(\mathcal{K})$ by finding appropriate

interpretations for the concepts and roles introduced by Ω . Thereby it is straightforward to extract a model for \mathcal{K} , given a model of $\Omega(\mathcal{K})$. In the remainder, we will assume a knowledge base in normalized form, if not stated otherwise.

4.1 Candidate Generation

As shown, any potential bounded interpretation $\mathcal{I}_{\mathbf{B}}$ is induced by a set of individual assertions \mathbf{B} , such that for each concept name A , role name r and individuals a, b occurring in \mathcal{K} , either $A(a) \in \mathbf{B}$, or $\neg A(a) \in \mathbf{B}$ and either $r(a, b) \in \mathbf{B}$ or $\neg r(a, b) \in \mathbf{B}$. This construction is straightforward to encode via subsequent rules:

$$\Pi_{\text{gen}}(\mathcal{K}) := \{A(X) :- \text{not } \neg A(X), \top(X) \mid A \in N_C(\mathcal{K})\} \cup \quad (1)$$

$$\{\neg A(X) :- \text{not } A(X), \top(X) \mid A \in N_C(\mathcal{K})\} \cup \quad (2)$$

$$\{ar(r, X, Y) :- \text{not } \neg ar(r, X, Y), \top(X), \top(Y) \mid r \in N_R(\mathcal{K})\} \cup \quad (3)$$

$$\{\neg ar(r, X, Y) :- \text{not } ar(r, X, Y), \top(X), \top(Y) \mid r \in N_R(\mathcal{K})\} \cup \quad (4)$$

$$\{\top(a) \mid a \in N_I(\mathcal{K})\}. \quad (5)$$

Recall, that a rule is *unsafe*, if a variable that occurs in the head does not occur in any positive body literal. The predicate $\top(X)$ ensures safe rules, each of the guessing rules (1–4) would otherwise be unsafe. This predicate represents the \top -concept, to which the statement (5) asserts each individual present in \mathcal{K} . The function ar takes care of potential inverse roles (cf. Table 4). Whereas “*not*” denotes default negation, \neg is without attached semantics and merely used as syntactic counterpart to the DL vocabulary. We show now that $\Pi_{\text{gen}}(\mathcal{K})$ computes $\mathcal{B}_{\mathcal{K}}$, the set of all constructible \mathbf{B} , and each $\mathbf{B} \in \mathcal{B}_{\mathcal{K}}$ determines a solution of $\Pi_{\text{gen}}(\mathcal{K})$.

Proposition 1 ($\mathcal{B}_{\mathcal{K}} = \mathcal{AS}(\Pi_{\text{gen}}(\mathcal{K}))$). *Let $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ be a *SRQIQ* knowledge base and $\Pi_{\text{gen}}(\mathcal{K})$ the logic program obtained by the translation given in (1–5). Then, it holds that $\mathcal{B}_{\mathcal{K}}$ coincides with the set of all answer sets of $\Pi_{\text{gen}}(\mathcal{K})$.*

4.2 Axiom Encoding

In the next step, we turn each axiom $\alpha \in \mathcal{R} \cup \mathcal{T}$ into a constraint, ultimately ruling out those candidate interpretations not satisfying α . Moreover, each individual assertion in the ABox \mathcal{A} restricts the search space further, since for some present fact $A(a)$ any solution candidate containing $\neg A(a)$ is eliminated. We will successively introduce appropriate encodings for axioms of each knowledge base component, altogether manifested in the program $\Pi_{\text{chk}}(\mathcal{K})$ and will finally show that the program $\Pi(\mathcal{K}) = \Pi_{\text{gen}}(\mathcal{K}) \cup \Pi_{\text{chk}}(\mathcal{K})$ computes all bounded models of \mathcal{K} .

Encoding TBox Axioms Since \mathcal{T} is normalized, each GCI is of certain form which simplifies the encoding. We obtain $\Pi_{\text{chk}}(\mathcal{T})$ as follows:

$$\Pi_{\text{chk}}(\mathcal{T}) := \{ :- \text{trans}(C_1), \dots, \text{trans}(C_n) \mid \text{for each } \top \sqsubseteq \bigsqcup_{i=1}^n C_i \text{ in } \mathcal{T} \} \quad (6)$$

Each concept expression C_i is translated according to the function $\text{trans}(C_i)$ depicted in Table 4. Note, each C_i is only one of the ones given in Definition 2, the ones given in the first column; i.e. not complex, with the nice effect of $\text{trans}(C_i)$ to be realized non-recursively.

Table 4. Translation of Concept Expressions.

C	$\text{trans}(C)$
A	$\text{not } A(X)$
$\neg A$	$A(X)$
$\{a\}$	$\{\text{not } O_a(X)\}, \{O_a(a)\}$
$\forall r.A$	$\{\text{not } A(Y_A), \text{ar}(r, X, Y_A)\}$
$\forall r.\neg A$	$\{\text{ar}(r, X, Y_A), A(Y_A)\}$
$\exists r.\text{Self}$	$\text{not } \text{ar}(r, X, X)$
$\neg \exists r.\text{Self}$	$\text{ar}(r, X, X)$
$\geq n \text{ r}.A$	$\#\text{count}\{\text{ar}(r, X, Y_A) : A(Y_A)\} < n$
$\geq n \text{ r}.\neg A$	$\#\text{count}\{\text{ar}(r, X, Y_A) : \text{not } A(Y_A)\} < n$
$\leq n \text{ r}.A$	$\#\text{count}\{\text{ar}(r, X, Y_A) : A(Y_A)\} > n$
$\leq n \text{ r}.\neg A$	$\#\text{count}\{\text{ar}(r, X, Y_A) : \text{not } A(Y_A)\} > n$

Note: O_a is a new concept name unique for a . And $\text{ar}(r, X, Y)$ is defined as follows:

$$\text{ar}(r, X, Y) := \begin{cases} r(X, Y) & \text{if } r \text{ is an atomic role} \\ s(Y, X) & \text{if } r \text{ is an inverse role and } r = s^- \end{cases}$$

Encoding RBox Axioms Role assertions and role inclusion axioms are also transformed into constraints, grouped in the program $\Pi_{\text{chk}}(\mathcal{R})$. According to their DL semantics, this yields:

$$\Pi_{\text{chk}}(\mathcal{R}) := \{ :- \text{ar}(r, X, Y), \text{not } \text{ar}(s, X, Y) \mid r \sqsubseteq s \in \mathcal{R} \} \cup \quad (7)$$

$$\{ :- \text{ar}(s, X, Y), \text{ar}(r, X, Y) \mid \text{Dis}(r, s) \in \mathcal{R} \} \cup \quad (8)$$

$$\{ :- \text{ar}(s_1, X, Y), \text{ar}(s_2, Y, Z), \text{not } \text{ar}(r, X, Z) \mid s_1 \circ s_2 \sqsubseteq r \in \mathcal{R} \}. \quad (9)$$

Encoding ABox Axioms The ABox \mathcal{A} itself represents an input database, which we can directly use. However, it remains to check whether \mathcal{A} does not contain contradictory knowledge; i.e. propositional clashes of the form $\{A(a), \neg A(a)\} \in \mathcal{A}$. Hence, the program $\Pi_{\text{chk}}(\mathcal{A})$ consists of \mathcal{A} and one additional constraint for each concept and role name ruling out inconsistent input ABoxes.

$$\Pi_{\text{chk}}(\mathcal{A}) := \mathcal{A} \cup \quad (10)$$

$$\{ :- A(X), \neg A(X) \mid A \in N_C(\mathcal{K}) \} \cup \quad (11)$$

$$\{ :- \text{ar}(r, X, Y), \neg \text{ar}(r, X, Y) \mid r \in N_R(\mathcal{K}) \}. \quad (12)$$

Note that the presence of $\{A(a), \neg A(a)\} \in \mathcal{A}$ does not cause an unsatisfiable program under the answer set semantics, since \neg does not have any meaning under the semantics; $\neg A$ is treated as just another predicate name. Thus, the imposed constraints simulate the known DL semantics.

Theorem 2. *Let $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ be a normalized \mathcal{SROIQ} knowledge base, and $\Pi(\mathcal{K}) = \Pi_{\text{gen}}(\mathcal{K}) \cup \Pi_{\text{chk}}(\mathcal{K})$ be the program obtained by applying Rules (1–12). Then, it holds:*

$$\mathcal{AS}(\Pi(\mathcal{K})) = \{\mathbf{B} \mid \mathbf{B} \in \mathcal{B}_{\mathcal{K}} \text{ and } \mathcal{I}_{\mathbf{B}} \models \mathcal{K}\}$$

With this theorem in place, we benefit from the translation in many aspects. Most notably, in addition to the standard DL reasoning tasks, *model extraction* and *model enumeration* can be carried out without additional efforts, since both are natural tasks for answer set solvers.

5 Evaluation

We implemented our approach as an open-source tool, named **Wolpertinger**.² The obtained logic programs can be evaluated with most modern ASP solvers. However, the evaluation was conducted using **Clingo** [6] for grounding and solving, since it currently is the most prominent solver leading the latest competitions [2]. We present preliminary evaluation results based on simple ontologies, encoding constraint-satisfaction-type combinatorial problems. Existing OWL ontologies typically used for benchmarking, e.g. SNOMED or GALEN [29,23], do not fit our purpose, since they are modeled with the classical semantics in mind and often have little or no ABox information.

Our tests provide runtimes of **Wolpertinger** and the popular **Hermit** reasoner [10]. Whereas a direct comparison would not be fair, the conducted tests shall merely show the feasibility of our approach and the infeasibility of the axiomatization using standard DL reasoners. The evaluation itself is conducted on a standard desktop machine.³

5.1 Unsatisfiability

We construct an unsatisfiable knowledge base $\mathcal{K}_n = (\mathcal{A}_n, \mathcal{T}_n, \emptyset)$, with \mathcal{T}_n and \mathcal{A}_n as follows:

$$\mathcal{T}_n = \{A_1 \sqsubseteq \exists r.A_2, \dots, A_n \sqsubseteq \exists r.A_{n+1}\} \cup \quad (13)$$

$$\{A_i \sqcap A_j \sqsubseteq \perp \mid 1 \leq i < j \leq n+1\} \quad (14)$$

$$\mathcal{A}_n = \{A_1(a_1), \top(a_1), \dots, \top(a_n)\} \quad (15)$$

Inspired by common pigeonhole-type problems, we have \mathcal{K}_n enforce an r -chain of length $n+1$ without repeating elements, yet, given only n individuals such a

² <https://github.com/wolpertinger-reasoner>

³ Unix operating system, 1.8 Ghz Intel Core i7 Processor, 4 GB memory. Both tools are executed with standard Java-VM settings.

Table 5. Runtime comparison for detecting unsatisfiability of \mathcal{K}_n .

#	Nominals	Concepts	Wolpertinger	HermiT	#Backtrackings
1	5	6	< 0,001 s	000,148 s	260
2	6	7	< 0,001 s	000,436 s	1.630
3	7	8	000,040 s	001,050 s	11.742
4	8	9	000,320 s	002,292 s	95.900
5	9	10	004,790 s	015,301 s	876.808
6	10	11	054,400 s	144,024 s	8.877.690
7	11	12	085,080 s	> 15 min	> 50×10^6

Table 6. Enumerating Sudoku Instances – Runtime Overview.

#	Models Requested	Time(Total)	Time(Solving)
1	100	006,014 s	000,130 s
2	1.000	006,372 s	000,560 s
3	10.000	008,558 s	002,800 s
4	100.000	034,096 s	027,960 s
5	1.000.000	279,059 s	273,150 s

model cannot exist. Table 5 depicts the runtimes for detecting unsatisfiability of \mathcal{K}_n , for increasing n . The durations correspond to the pure solving time of **Clingo** and pure reasoning time of **HermiT**, respectively, as both **Wolpertinger** and **HermiT** have a comparable preprocessing. As the figures suggest, \mathcal{K}_n is a potential worst-case scenario, where both tools are doomed to test all combinations. On this task, **Wolpertinger** constantly outperforms **HermiT**. For \mathcal{K}_{11} , **HermiT** is stopped after 15 minutes, whereas **Wolpertinger** detects unsatisfiability within 85 seconds.

5.2 Model Extraction and Model Enumeration

With Table 6, we next provide some figures for model extraction and partial enumeration (retrieving a given number of bounded models). To this end, we created a knowledge base modeling fully and correctly filled Sudokus, featuring 108 named individuals, 13 concept names and 1 role name. When invoking a satisfiability test on this knowledge base using **HermiT**, no answer was given within 15 minutes.

On average, **Wolpertinger** provides a solution for a given Sudoku instance in around 6 seconds, of which more than 5 seconds are needed for grounding, while the actual solving is done in less than 0.1 seconds. For model enumeration, we used the knowledge base but removed information concerning pre-filled cells, turning the task into generating new Sudoku instances. The size of the grounded program is 13 MB, the grounding process taking around 6 seconds as reflected in Table 6.

6 Conclusion

With this paper, we have established the starting point for further developments on the theoretical and practical side, as well as we can identify benefits for both, the description logic and logic programming community. For the latter, our approach enables one to use OWL as ASP modeling language and therefore make use of the available tool support. Although modeling features are limited, we argue that quite large and involved problem scenarios can be modeled in OWL ontologies. Clearly, evaluations of our system with respect to such ontologies remain as imperative issue.

Complementarily, model extraction and enumeration supplement DL reasoning tasks for which our ASP translation represents not only a feasible approach, but apparently also a use case of ASP in another research field. Moreover, the framework may be extended to realize non-standard reasoning tasks useful for debugging purposes such as axiom pinpointing, explanation, justification and abduction, exploiting the innate capabilities of ASP to realize minimization as well as model enumeration.

On a more practical level, the proposed translation can certainly be optimized to exploit more built-in features of today’s ASP solvers. In terms of harnessing the convenience of OWL modeling environments, we will implement an OWL API reasoner interface for *Wolpertinger*, such that it can e.g., be seamlessly be integrated with other OWL software, such as Protégé [16].

Regarding future theoretical DL investigations, in recent years, significant extensions of the modeling and querying capabilities of DLs have been proposed and partially implemented. A major such extension is considering the reasoning task of answering queries, most prominently (unions of) conjunctive queries, positive queries, conjunctive 2-way regular path queries, and monadically defined queries subsuming all of the former [27]. It is not overly difficult to show that answering all these query types over *SRQIQ* knowledge bases (and hence over OWL ontologies) under the bounded model semantics is Π^2_P -complete, which again contrasts with the much worse results (if any) for the unbounded case [26,11]. Moreover, as all these query formalisms can be straightforwardly expressed in a rule-based way, an integration in our framework is immediate. In the same way, rule-based extensions of OWL – monotonic [14,21] or nonmonotonic [20,1] – should be straightforward to accommodate, at the cost of the combined complexity jumping to EXPTIME or NEXPTIME.

Acknowledgements

We are grateful for all the valuable feedback from our colleagues and the anonymous workshop reviewers, which helped greatly to improve this work.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Query-Driven Procedures for Hybrid MKNF Knowledge Bases. *ACM Transactions on Computational Logic* 14(2), 16:1–16:43 (2013)
2. Alviano, M., Calimeri, F., Charwat, G., Dao-Tran, M., Dodaro, C., Ianni, G., Krennwallner, T., Kronegger, M., Oetsch, J., Pfandler, A., Pührer, J., Redl, C., Ricca, F., Schneider, P., Schwengerer, M., Spendier, L.K., Wallner, J.P., Xiao, G.: The Fourth Answer Set Programming Competition: Preliminary Report. In: Cabalar, P., Son, T.C. (eds.) *Proc. of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013)*. LNCS, vol. 8148, pp. 42–53. Springer (2013)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edn. (2007)
4. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. *Commun. ACM* 54(12), 92–103 (2011)
5. Calvanese, D.: Finite model reasoning in description logics. In: *Proc. of the International Workshop on Description Logics (DL 1996)*. pp. 25–36 (1996)
6. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam Answer Set Solving Collection 24(2), 107–124 (2011)
7. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1–238 (2012)
8. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K.A. (eds.) *Proc. of the 5th International Conference and Symposium on Logic Programming*. pp. 1070–1080. MIT Press (1988)
9. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Comput.* 9(3/4), 365–386 (1991)
10. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: an OWL 2 reasoner. *Journal of Automated Reasoning* 53(3), 245–269 (2014)
11. Glimm, B., Kazakov, Y., Lutz, C.: Status QIO: An Update. In: Rosati, R., Rudolph, S., Zakharyashev, M. (eds.) *Proc. of the 24th International Workshop on Description Logics (DL 2011)*. CEUR Workshop Proceedings, vol. 745. CEUR-WS.org (2011)
12. Horridge, M., Patel-Schneider, P.F.: *OWL 2 Web Ontology Language Manchester Syntax* (2012)
13. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible *SROIQ*. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) *Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*. pp. 57–67. AAAI Press (2006)
14. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosz, B.N., Dean, M.: SWRL: A Semantic Web Rule Language. W3C Member Submission (21 May 2004)
15. Kazakov, Y.: *RIQ* and *SROIQ* are harder than *SHOIQ*. In: Brewka, G., Lang, J. (eds.) *Proc. of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*. pp. 274–284. AAAI Press (2008)
16. Knublauch, H., Musen, M.A., Rector, A.L.: Editing Description Logic Ontologies with the Protégé OWL Plugin. In: Haarslev, V., Möller, R. (eds.) *Proc. of the International Workshop on Description Logics (DL 2004)*. CEUR Workshop Proceedings, vol. 104. CEUR-WS.org (2004)

17. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. *ACM Trans. Comput. Log.* 7(3), 499–562 (2006)
18. Lutz, C., Sattler, U., Tendera, L.: The complexity of finite model reasoning in description logics. *Information and Computation* 199(1-2), 132–171 (May 2005)
19. Mitchell, D.G., Ternovska, E.: A framework for representing and solving NP search problems. In: *Proc. of AAAI*. vol. 5, pp. 430–435 (2005)
20. Motik, B., Rosati, R.: Reconciling description logics and rules. *Journal of the ACM* 57(5) (2010)
21. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL DL with Rules. *Journal of Web Semantics* 3(1), 41–60 (2005)
22. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. *Artificial Intelligence Research* 36, 165–228 (2009)
23. Rector, A., Horrocks, I.: Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In: *Proc. of the Workshop on Ontological Engineering* (1997)
24. Rosati, R.: Finite Model Reasoning in DL-Lite. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *Proc. of the 5th European Semantic Web Conference on The Semantic Web: Research and Applications (ESWC 2008)*. LNCS, vol. 5021, p. 215. Springer (2008)
25. Rudolph, S.: Foundations of Description Logics. In: Polleres, A., d’Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P.F. (eds.) *Reasoning Web. 7th International Summer School 2011, Tutorial Lectures*. LNCS, vol. 6848, pp. 76–136. Springer (2011)
26. Rudolph, S., Glimm, B.: Nominals, Inverses, Counting, and Conjunctive Queries or: Why Infinity is your Friend! *Journal of Artificial Intelligence Research* 39, 429–481 (2010)
27. Rudolph, S., Krötzsch, M.: Flag & Check: Data Access with Monadically Defined Queries. In: Hull, R., Fan, W. (eds.) *Proc. of the 32nd Symposium on Principles of Database Systems (PODS’13)*. pp. 151–162. ACM (2013)
28. Savitch, W.J.: Relationships Between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci.* 4(2), 177–192 (Apr 1970)
29. Spackman, K.A., Campbell, K.E., Côté, R.A.: SNOMED RT: a reference terminology for health care. In: *AMIA 1997, American Medical Informatics Association Annual Symposium*. AMIA (1997)
30. Stockmeyer, L.J.: The Complexity of Decision Problems in Automata Theory and Logic. Ph.D. thesis, Massachusetts Institute of Technology (1974)
31. W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation, <http://www.w3.org/TR/owl2-overview> (2009)

A Proofs

Proof of Theorem 2 By Proposition 1, $\Pi_{\text{gen}}(\mathcal{K})$ computes the set $\mathcal{B}_{\mathcal{K}}$. It remains to show, that $\Pi_{\text{chk}}(\mathcal{K})$ obeys the bounded model semantics, and consequently excludes each $\mathbf{B} \in \mathcal{B}_{\mathcal{K}}$ not inducing a bounded model $\mathcal{I}_{\mathbf{B}}$.

$\mathcal{AS}(\Pi(\mathcal{K})) \subseteq \{\mathbf{B} \mid \mathbf{B} \in \mathcal{B}_{\mathcal{K}} \text{ and } \mathcal{I}_{\mathbf{B}} \models \mathcal{K}\}$ Let $\mathbf{I} \in \mathcal{AS}(\Pi(\mathcal{K}))$ be an answer set of $\Pi(\mathcal{K})$. From Proposition 1, we know $\mathbf{I} \in \mathcal{B}_{\mathcal{K}}$. We show now that the interpretation $\mathcal{I}_{\mathbf{I}}$ induced by \mathbf{I} is a bounded model of \mathcal{K} , and therefore $\mathcal{I}_{\mathbf{I}} \models \alpha$, for each axiom $\alpha \in \mathcal{K}$. Then, let

$\alpha \in \mathcal{R}$: we distinguish role disjointness, and role inclusion axioms:

$\alpha \in \mathcal{R}_a$: Let $\alpha = \text{Dis}(r, s) \in \mathcal{R}_a$, then by definition of $\Pi_{\text{chk}}(\mathcal{R})$, there is a ground constraint $\rho_{\alpha} = :- s(a, b), r(a, b)$ in $\text{Gr}(\Pi_{\text{chk}}(\mathcal{R}))$, for all individuals $a, b \in N_I(\mathcal{K})$. Since \mathbf{I} is an answer set, $\{s(a, b), r(a, b)\} \notin \mathbf{I}$. Consequently either $(a, b) \in s^{\mathcal{I}_{\mathbf{I}}}$, or $(a, b) \in r^{\mathcal{I}_{\mathbf{I}}}$, hence $\mathcal{I}_{\mathbf{I}} \models \text{Dis}(r, s)$.

$\alpha \in \mathcal{R}_h$: then let α be of the form $s_1 \circ s_2 \sqsubseteq r$, with $s_1, s_2, r \in N_R(\mathcal{K})$, and $\rho_{\alpha} = :- s_1(a_1, a_2), s_2(a_2, a_3)$, *not* $r(a_1, a_3)$ be the ground constraint in $\text{Gr}(\Pi_{\text{chk}}(\mathcal{R}))$. Since \mathbf{I} is an answer set, we have that, if $s_1(a_1, a_2)$ and $s_2(a_2, a_3) \in \mathbf{I}$ implies $r(a_1, a_3) \in \mathbf{I}$. And consequently $(a_1, a_2) \in s_1^{\mathcal{I}_{\mathbf{I}}}$, $(a_2, a_3) \in s_2^{\mathcal{I}_{\mathbf{I}}}$ and $(a_1, a_3) \in r^{\mathcal{I}_{\mathbf{I}}}$, thus $\mathcal{I}_{\mathbf{I}} \models s_1 \circ s_2 \sqsubseteq r$.

$\alpha \in \mathcal{T}$: then α is normalized and of the form $\top \sqsubseteq \bigsqcup_{i=1}^n C_i$. In Rule (6), α is turned into a constraint $\rho_{\alpha} = :- \text{trans}(C_1), \dots, \text{trans}(C_n)$ in $\Pi_{\text{chk}}(\mathcal{T})$. Since \mathbf{I} is an answer set, it does not violate any of the grounded instances of ρ_{α} in $\text{Gr}(\Pi_{\text{chk}}(\mathcal{T}))$. Suppose now towards contradiction, $\mathcal{I}_{\mathbf{I}}$ induced by \mathbf{I} does not satisfy α , $\mathcal{I}_{\mathbf{I}} \not\models \alpha$. Then, $\mathcal{I}_{\mathbf{I}} \not\models C_i$, for all $1 \leq i \leq n$. However, since \mathbf{I} does not violate ρ_{α} , in each of the ground instantiations of ρ_{α} , there is exists a $\text{trans}(C_i)$ which is not satisfied by \mathbf{I} , $1 \leq i \leq n$. Then, C_i is one of the expressions given in Definition 2, and we distinguish:

$C_i = A$: then $\text{trans}(C_i) = \text{not } A(X)$, and $A(a) \in \mathbf{I}$ for any $a \in N_I(\mathcal{K})$. Consequently $a \in A^{\mathcal{I}_{\mathbf{I}}} \neq \emptyset$, which contradicts the assumption $\mathcal{I}_{\mathbf{I}} \not\models C_i$.

$C_i = \neg A$: then $\text{trans}(C_i) = A(X)$, and $\neg A(a) \in \mathbf{I}$ for any $a \in N_I(\mathcal{K})$. Consequently $a \in (\neg A)^{\mathcal{I}_{\mathbf{I}}} \neq \emptyset$, which contradicts the assumption $\mathcal{I}_{\mathbf{I}} \not\models C_i$.

$C_i = \{a\}$: then $\text{trans}(C_i) = \{\text{not } O_a(X)\}$ and $O_a(a)$, thus necessarily $O_a(a) \in \mathbf{I}$. In order to not satisfy $\text{trans}(C_i)$, $X = a$. Consequently we have $a \in O_a^{\mathcal{I}_{\mathbf{I}}}$ with O_a as nominal guard concept, and therefore $\{a\}^{\mathcal{I}_{\mathbf{I}}} = \{a\}$, which contradicts the assumption.

$C_i = \forall r.A$: then $\text{trans}(C_i) = \{r(X, Y_A), \text{not } A(Y_A)\}$, and $A(b) \in \mathbf{I}$ whenever $r(a, b) \in \mathbf{I}$. Consequently, $(a, b) \in r^{\mathcal{I}_{\mathbf{I}}}$ implies $b \in A^{\mathcal{I}_{\mathbf{I}}}$, which contradicts the assumption $\mathcal{I}_{\mathbf{I}} \not\models C_i$.

$C_i = \geq n r.A$: then $\text{trans}(C_i) = \#\text{count}\{r(X, Y_A) : A(Y_A)\} < n$, and more or equal than n , say m , atoms $r(a, b) \in \mathbf{I}$ and $A(b) \in \mathbf{I}$. Consequently we find also m pairs $(a, b) \in r^{\mathcal{I}_{\mathbf{I}}}$ and $b \in A^{\mathcal{I}_{\mathbf{I}}}$, which contradicts the assumption.

All remaining cases can be treated analogously.

$\alpha \in \mathcal{A}$: \mathbf{I} satisfies $\Pi_{\text{chk}}(\mathcal{A})$, in particular $\mathcal{A} \subseteq \mathbf{I}$. Moreover, none of the imposed constraints in $\Pi_{\text{chk}}(\mathcal{A})$ is violated, proving consistency of \mathcal{A} , and therefore $\{\mathbf{A}(a), \neg\mathbf{A}(a)\} \notin \mathbf{I}$ and $\{\mathbf{r}(a, b), \neg\mathbf{r}(a, b)\} \notin \mathbf{I}$ for all concept names $A \in N_C(\mathcal{K})$, role names $r \in N_R(\mathcal{K})$ and individuals $a, b \in N_I(\mathcal{K})$.

$\mathcal{AS}(\Pi(\mathcal{K})) \supseteq \{\mathbf{B} \mid \mathbf{B} \in \mathcal{B}_{\mathcal{K}} \text{ and } \mathcal{I}_{\mathbf{B}} \models \mathcal{K}\}$ Let $\mathcal{I}_{\mathbf{B}}$ be a bounded model of \mathcal{K} , induced by some $\mathbf{B} \in \mathcal{B}_{\mathcal{K}}$. We show that \mathbf{B} is an answer set of $\Pi(\mathcal{K}) = \Pi_{\text{gen}}(\mathcal{K}) \cup \Pi_{\text{chk}}(\mathcal{K})$. From Proposition 1 we know, that \mathbf{B} is an answer set of $\Pi_{\text{gen}}(\mathcal{K})$, thus it remains to show that \mathbf{B} satisfies $\Pi_{\text{chk}}(\mathcal{K})$ and therefore does not violate any of the imposed constraints. Since $\mathcal{I}_{\mathbf{B}} \models \alpha$, for each $\alpha \in \mathcal{K}$, let

$\alpha \in \mathcal{R}$: we distinguish again role disjointness and role chain axioms.

$\alpha \in \mathcal{R}_a$: Let $\alpha = \text{Dis}(r, s) \in \mathcal{R}_a$ and $\rho_\alpha = \mathbf{r}(X, Y), \mathbf{s}(X, Y)$, be the constraint according to Rule (8). Since $\mathcal{I}_{\mathbf{B}} \models \alpha$, for all $a, b \in N_I(\mathcal{K})$ we find, that $\{\mathbf{r}(a, b), \mathbf{s}(a, b)\} \notin \mathbf{B}$, and consequently none of the grounded instances of ρ_α is violated by \mathbf{B} .

$\alpha \in \mathcal{R}_h$: then α is of the form $s_1 \circ s_2 \sqsubseteq r$, with $s_1, s_2, r \in N_R(\mathcal{K})$, and $\rho_\alpha = :- \mathbf{s}_1(X, Y_1), \mathbf{s}_2(Y_n, Z), \text{not } \mathbf{r}(X, Z)$ is the constraint according to Rule (9). Since $\mathcal{I}_{\mathbf{B}} \models \alpha$, for all $a_1, a_2, a_3 \in N_I(\mathcal{K})$ we have that if $s_1(a_1, a_2)$ and $s_2(a_2, a_3) \in \mathbf{B}$, then $\mathbf{r}(a_1, a_3) \in \mathbf{B}$. Consequently, ρ_α is not violated by \mathbf{B} .

$\alpha \in \mathcal{T}$: then α is normalized and of the form $\top \sqsubseteq \bigsqcup_{i=1}^n C_i$, which is satisfied by $\mathcal{I}_{\mathbf{B}}$, iff $C_i^{\mathcal{I}_{\mathbf{B}}} \neq \emptyset$ for some $1 \leq i \leq n$. Let $\rho_\alpha = :- \text{trans}(C_1), \dots, \text{trans}(C_n)$ be the constraint obtained from α , applying Rule (6). We need to show that \mathbf{B} does not violate the constraint. Let C_i be the concept expression for which $C_i^{\mathcal{I}_{\mathbf{B}}} \neq \emptyset$ holds, $1 \leq i \leq n$, and C_i is one of the expressions given in Definition 2, in particular we have:

$C_i = A$: then $A^{\mathcal{I}_{\mathbf{B}}} = \{a \mid \mathbf{A}(a) \in \mathbf{B}\}$. Consequently, for each of those $\mathbf{A}(a) \in \mathbf{B}$, $\text{trans}(A) = \text{not } \mathbf{A}(X)$ is not satisfied.

$C_i = \neg A$: then $\neg A^{\mathcal{I}_{\mathbf{B}}} = \{a \mid \neg\mathbf{A}(a) \in \mathbf{B}\}$. Consequently, for each of those $\neg\mathbf{A}(a) \in \mathbf{B}$, $\text{trans}(\neg A) = \mathbf{A}(X)$ is not satisfied.

$C_i = \exists r. \text{Self}$: then $(\exists r. \text{Self})^{\mathcal{I}_{\mathbf{B}}} = \{a \mid \mathbf{r}(a, a) \in \mathbf{B}\}$. Consequently, $\text{trans}(\exists r. \text{Self}) = \text{not } \mathbf{r}(X, X)$ is not satisfied for those $\mathbf{r}(a, a) \in \mathbf{B}$.

$C_i = \leq r n. A$: then $(\leq r n. A)^{\mathcal{I}_{\mathbf{B}}} = \{a \mid \#\{\mathbf{r}(a, b) \text{ and } \mathbf{A}(b) \in \mathbf{B}\} = m \leq n\}$. Consequently, $\text{trans}(\leq r n. A) = \# \text{count}\{\mathbf{r}(X, Y_A) : \mathbf{A}(Y_A)\} > n$, is not satisfied since there are m such $\mathbf{r}(a, b) \in \mathbf{B}$ with $\mathbf{A}(b) \in \mathbf{B}$.

All remaining cases can be treated analogously.

$\alpha \in \mathcal{A}$: then $\alpha \in \mathbf{B}$, as well as $\alpha \in \Pi_{\text{chk}}(\mathcal{A})$, since by definition $\mathcal{A} \in \Pi_{\text{chk}}(\mathcal{A})$. In general, since $\mathcal{I}_{\mathbf{B}} \models \mathcal{K}$, \mathcal{A} is consistent and therefore $\{\mathbf{A}(a), \neg\mathbf{A}(a)\} \notin \mathbf{B}$, as well as $\{\mathbf{r}(a, b), \neg\mathbf{r}(a, b)\} \notin \mathbf{B}$ for all concept names A , role names r and individuals a, b . \square